

# SimTensor: A synthetic tensor data generator

**Hadi Fanaee-T**

*INESC TEC*

*Campus of FEUP*

*4200-465 Porto, Portugal*

HADI.F.TORK@INESCTEC.PT

**Joao Gama**

*INESC TEC*

*Campus of FEUP*

*4200-465 Porto, Portugal*

JGAMA@FEP.UP.PT

## Abstract

SimTensor is a multi-platform, open-source software for generating artificial tensor data (either with CP/PARAFAC or Tucker structure) for reproducible research on tensor factorization algorithms. SimTensor is a stand-alone application based on MATAB. It provides a wide range of facilities for generating tensor data with various configurations. It comes with a user-friendly graphical user interface, which enables the user to generate tensors with complicated settings in an easy way. It also has this facility to export generated data to universal formats such as CSV and HDF5, which can be imported via a wide range of programming languages (C, C++, Java, R, Fortran, MATLAB, Perl, Python, and many more). The most innovative part of SimTensor is this that can generate temporal tensors with periodic waves, seasonal effects and streaming structure. it can apply constraints such as non-negativity and different kinds of sparsity to the data. SimTensor also provides this facility to simulate different kinds of change-points and inject various types of anomalies. The source code and binary versions of SimTensor is available for download in <http://www.simtensor.org>.

**Keywords:** Tensor factorization, CP, Tucker, data generator, synthetic data

## 1. Introduction

Tensor factorizations have lots of applications in data mining, machine learning, and signal processing as well as chemometrics, and computer vision (Kolda and Bader, 2009; Mørup, 2011; Fanaee-T and Gama, 2016b; Papalexakis et al., 2016). Obviously any improvement in the accuracy of tensor factorization algorithms advances the state-of-the-art in the above-mentioned domains. Developing accurate algorithms for tensor factorization require benchmark data sets with various realistic adjustable configurations. One of the characteristics that is not included in many existing tensor data generators is time-changing behavior of realistic tensors as well as many real issues such as change-points and anomalies. The objective of SimTensor is to provide a standard and comprehensive framework for generating synthetic tensor structured data with focus on the time-changing characteristics of data. The SimTensor is developed based upon of many available codes and techniques in the literature of tensor analysis. So it is benefited a lot from the open-source contributions. Our methodology was to carefully study the available features and considered issues

in the existing data generators and the presented approaches in the literature for creating test problems. Besides, some ideas such as injecting anomalous tensor is being presented for the first time in SimTensor and is not reported elsewhere. Therefore, SimTensor can be considered as the state-of-the-art tool for generating synthetic tensor data with either CP (Carroll and Chang, 1970; Carroll et al., 1980) or Tucker (Tucker, 1966) structure with focus on time-changing tensors that are very realistic in many applications.

The existing data generators featured in toolboxes such as Tensor Toolbox (Bader et al., 2015) and N-Way Toolbox (Andersson and Bro, 2000) are presented as functions in MATLAB. Hence the access of users of other programming environments is limited since MATLAB is a licensed software. The SimTensor provides a stand-alone solution for wider range of practitioners who develop tensor factorization algorithms in various programming languages and operating systems and want to test their algorithms with several realistic configurations. Therefore, it is expected that SimTensor can contribute to reproducible research on tensor factorization algorithms. It is also open-source, so the community can actively contribute to the project and add their desired features to it according to the requirements that are not considered in the current version.

## 2. Generating Non-temporal Random Factors

One of the basic step to generate synthetic tensors with CP structure is to create random factor matrices  $U^{(n)}$  and then compute sum of multiway-way outer products:

$$x_{ijk} = \sum_{r=1}^R u_{ir}^{(1)} u_{jr}^{(2)} u_{kr}^{(3)}. \quad (1)$$

Where  $R$  is the number of components,  $I, J, K$  represent the size of tensor in each dimension, and  $N$  is order of tensor. If we assume that columns of  $U^{(n)}$  are normalized we can introduce the vector  $\lambda \in \mathbb{R}^R$  and re-write (1) as:

$$x_{ijk} = \sum_{r=1}^R \lambda_r u_{ir}^{(1)} u_{jr}^{(2)} u_{kr}^{(3)}. \quad (2)$$

For the Tucker tensor in addition to factor matrices we have to generate a random core tensor  $\mathcal{G}$  as well. For instance, for a third-order tensor we have:

$$x_{ijk} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} u_{ir_1}^{(1)} u_{jr_2}^{(2)} u_{kr_3}^{(3)}. \quad (3)$$

In order to generate  $U^{(n)}$  we can use different approaches. These techniques will be described in the following subsections.

### 2.1 Gamma Distribution

In SimTensor, the module *gamma* generates random factors from  $\Gamma(k, \theta)$  where  $\Gamma$  is Gamma function,  $k$  is positive scalar value called shape parameter, and  $\theta$  is nonnegative scalar value called scale parameter. As (Hu et al., 2015) in SimTensor we choose  $k$  from  $|\mathcal{N}(\mu, \sigma^2)|$

where  $\mathcal{N}$  is Gaussian distribution,  $\mu$  is mean, and  $\sigma^2$  is variance. In SimTensor, by default  $\mu$  and  $\sigma^2$  are set as 0.1 and  $\theta$  is set as 0.01.

## 2.2 Multivariate Gaussian Distribution

The module *multi\_normal\_dist* allows to generate  $R$  columns in factor matrix via multivariate Gaussian distribution with different  $\mu$  and  $\sigma$  parameters (Zhao et al., 2016) for each column. In SimTensor by default  $\mu$  and  $\sigma$  are set as 0 and 1, for all columns.

## 2.3 Uniformly Distributed Random Numbers

The module *rand* in SimTensor generates random factors with uniformly distributed random numbers on  $[0,1]$ . This method is widely used for generating random factors, for instance in Tensor toolbox (Bader et al., 2015) and elsewhere.

## 2.4 Standard Normal Distribution

The module *randn* generate factor matrices drawn from the standard normal distribution with  $\mu = 0$  and  $\sigma = 1$  for all columns. Note that in the current version of SimTensor *randn* is identical to module *multi\_normal\_dist*. Because by default we set  $\mu = 0$  and  $\sigma = 1$  in all columns for both methods. However, *multi\_normal\_dist* opposed to *randn* has this ability to generate columns in factor matrices with different  $\mu$  and  $\sigma$ . The *randn* module is also used in Tensor toolbox (Bader et al., 2015) for creating test problems.

## 2.5 Random Orthogonal Matrices

The module *orthogonal* is borrowed from Tensor toolbox (Bader et al., 2015) and generates a random  $n \times n$  orthogonal matrix, a matrix distribution uniform over the manifold of orthogonal matrices with respect to the induced  $\mathbb{R}^{n^2}$  Lebesgue measure (Shilon, 2016).

## 2.6 Stochastic

The module *stochastic* as its equivalent in Tensor toolbox (Bader et al., 2015) generates nonnegative factor matrices so that each column sums to one. This method works as follow. First, a factor matrix of  $U^n \in \mathbb{R}^{I^n \times R^n}$  is generated via uniformly distributed random numbers on  $[0,1]$ . Then the sum of each column of  $U^n$  is obtained and stored in vector  $S$ . Final generated matrix is obtained by multiplying  $U^n$  to diagonal matrix of  $S^{-1}$ .

## 2.7 Binary factors

The idea of binary factors is borrowed from work of (Sakurai, 2016) that creates boolean synthetic tensor by generating random binary factors. The method is very simple. First, we generate matrix of  $U^n \in \mathbb{R}^{I^n \times R^n}$  filled with zero. Then for each row of matrix we randomly fill one column of matrix with one, such that in each row we will have only one column with one.

## 2.8 Random Core Tensor/Lambda Generator

SimTensor allows the user to generate random core/tensor (for Tucker) and lambda (for CP). The methods include vector of ones (ones), uniformly distributed random numbers (rand), and standard normal distribution (randn). The user also can customize the generated vector. For Tucker case the generated vector is automatically transformed to the core tensor.

## 3. Generating Temporal Random Factors

SimTensor enables the user to create temporal factors with different strategies such as periodic waves, seasonal effects, and streaming.

### 3.1 Periodic waves

In SimTensor is possible to simulate periodic waves such as Cosine, Sine, as well as Square and Sawtooth waves. The user is able to determine the number of waves and frequency of waves in each setting. The idea is inspired by (Lemm et al., 2011) for generating artificial data for testing algorithms for factorization of shift-invariant multilinear data which is a real case in neuroimaging data.

### 3.2 Seasonal effects

Seasonal effects are very realistic for those data sets that are generated by humans. In such data sets there is a repeating pattern during specific time interval such as days, weeks, seasons, and so forth. For instance, in disease surveillance data (Fanaee-T and Gama, 2015a) disease outbreaks such as Influenza are more frequent in winters. In computer networks different traffic usage pattern can be detected. For instance, it is observed that traffic peak occurs around mid-day and during the night (Tune and Roughan, 2013). In transportation systems the peak of traffic is on the morning when people go to work and evening when they back home (Tan et al., 2013). There are lots of examples of such temporal seasonality. In SimTensor is possible to simulate different seasonal effects with various temporal granularities. Sometimes also in some other applications such as animal migration we may find some specific seasonality patterns which can be different from human-centered systems. In SimTensor the user is able to define some customized seasonal effects like this as well. The user also is able to define a particular growth rate. In this case, the data items in each cycle will be multiplied by a weight.

As in many realistic scenarios (e.g. in transportation systems) also we may experience multiple seasonality patterns in data. It is extremely easy to simulate such scenarios in SimTensor with defining multiple factors with different seasonal grounds.

### 3.3 Streaming tensor

In the approach that is exploited in (Nion and Sidiropoulos, 2009) for generating time-varying factor matrices it is assumed that data arrives in a streaming fashion and it is expected that data in  $t+1$  is a sample of data of  $t$  with some little changes. This change is controlled by a parameter that is called *variation control parameter* and can be defined by

the user. This can be used for evaluation of any incremental or streaming tensor analysis algorithm (Sun et al., 2008; Zhou et al., 2016; Fanaee-T and Gama, 2015b).

#### 4. Simulating Change Points

Shifts and drifts are realistic characteristics in many applications. It is important to assess the performance of tensor factorization algorithms in dealing with such changes. In SimTensor it is possible to create artificial change points in the factor matrices related to the temporal dimension. Depending on the defined period by the user different types of changes can be simulated, including temporary changes, structural shifts, and singular outliers. For instance, if temporal mode has size of 100, the user can simulate a shift by defining the change point with start point of 51 and end point of 100. Example of short-term changes (or events) can be start point of 20 and end point of 25. And finally if user is interested to simulate singular changes (or temporal outliers) she can define start/end point at the same time instant.

#### 5. Simulating Anomalies

Those patterns in data that do not conform to expected behavior are called anomalies (Chandola et al., 2009; Fanaee-T and Gama, 2016b). In different contexts anomalies may have diverse interpretations. In medical domain, for instance it can be translated to disease outbreak. In industrial settings it might be a fault. Or in transportation systems it might be translated as a events (Fanaee-T and Gama, 2016a). However, anomalies are different from outliers, in the sense that anomalies occur in a systematic way. In fact anomalies are generated by anomalous process while outliers can be only small errors in information systems or data gathering (Fanaee-T et al., 2014). In SimTensor we for the first time propose a new method for simulating anomalies. The idea is that we first create a smaller random tensor with CP structure and then inject it inside the bigger generated tensor. Actually we replace a small portion of the original tensor with the new generated tensor. This will be a challenge for tensor factorization algorithms to discover this small injected tensor. It is assumed that those approaches that can detect the smaller tensor should perform well for detecting anomalies from tensor-structured data.

#### 6. Simulating Noise

Noise is the inherent part of many data sets. Three methods are available in SimTensor for simulating noises. The user is able to apply noise directly to the factor matrices; apply an additive white Gaussian noise with controllable noise level on the final generated tensor (more common) via the method of (Viswanathan, 2015); or apply sparse white noise on the final tensor.

#### 7. Constraints on factor matrices

SimTensor enables the user to add non-negativity constraint in two ways, either on factor matrices or on the final tensor. The user can also apply different constraints such as correlation between columns in the factor matrices or angle between

Component	Module	Target tensor factorization algorithm	Target Data
Non-temporal factor generator	rand, randn, multi_normal_dist	Traditional approaches, for example CP-ALS (Carroll and Chang, 1970; Carroll et al., 1980) or Tucker-ALS (Tucker, 1966)	High quality multi-linear data
	Orthogonal	HOSVD (De Lathauwer et al., 2000b), HOOI (De Lathauwer et al., 2000a) or similar	High quality multi-linear data
	Stochastic	Non-negative algorithms like (Carroll et al., 1989; Kim and Choi, 2007)	Visual data (e.g. image or video) and count data (e.g. recommender systems)
	Binary	Boolean algorithms such as (Miettinen, 2011)	Social network data
Temporal factor generator	Periodic waves	Shift-invariant TD algorithms like ShiftCP (Mørup et al., 2008)	Neuroimaging data (e.g. EEG)
	Seasonal effects	All Algorithms	Human generated data sets (e.g. Internet traffic, mobility data, etc.)
	Streaming	Online Algorithms like PARAFAC-SDT (Nion and Sidiropoulos, 2009), OnlineCP (Zhou et al., 2016) or similar	Streaming data
	Change-points	Incremental algorithms such as DTA, STA, or WTA (Sun et al., 2008)	Temporal tensors
Effects	Anomalies	All algorithms (with focus on the evaluation of model's discriminability)	Data sets with structural anomalies
	Noise	All algorithms (with focus on the evaluation of model's power)	All Data sets
	Non-negativity constraints	Non-negative algorithms like (Carroll et al., 1989; Kim and Choi, 2007)	Visual data (e.g. image or video) and count data (e.g. recommender systems)
	Sparsity constraints	Sparse-friendly algorithms such as (Kolda and Sun, 2008)	Incomplete data sets
	Sparse Count Tensors	Sparse Bayesian algorithms such as (Hu et al., 2015)	Large-scale sparse count data in recommender systems and social networks

Table 1: A guide to choose the the component and module in SimTensor for evaluation of a specific tensor factorization algorithm or testing an algorithm in a particular application.

columns of the factor matrices. Note that this can be applied to non-temporal modes. It is also possible to normalize the final tensor or perform a sign fix operation on the final tensor.

## 8. Generating Sparse tensors

Three mechanisms are included for generating sparse tensors. The first one is to apply sparsity constraint on the factor matrices by randomly removing non-zero elements from the dense created factor matrices. The second strategy is to generate CP or Tucker tensor with the dense factor matrices and then remove some random non-zero elements. The third methodology is to create sparse count tensors based on the idea of (Hu et al., 2015). In this method we firstly generate random factors with Gamma random numbers. Then we create CP tensor with the generated random factors and finally feed this tensor as input parameters for generating tensor with Poisson distribution. This kind of synthetic data can be used for evaluation of Bayesian tensor factorization algorithms (Hu et al., 2015).

## 9. How to use SimTensor?

Depending on the evaluation objective SimTensor can generate synthetic data for various family of tensor factorization algorithms. It also can create data sets with different realistic characteristics that exists in many applications. In Table 1 a guide is presented to choose the right module and component for generating synthetic tensor when any specific type of algorithms or data sets is desired.

## Acknowledgments

We would like to acknowledge support for this project from the European Comission (EU FP7 grant ICT-2013-612944). The development of SimTensor has benefited from the tensor analysis community, including the works and the released codes by: Andrzej Cichocki, Anh-Huy Phan, Arun Tejasvi Chaganty, Brett W. Bader, Changwei Hu, Changyou Chen, Christos Faloutsos, Claus A. Andersson, Daniel M. Dunlavy, Dimitri Nion, Eric C. Chi, Evrim Acar, Giorgio Tomasi, Guoxu Zhou, Haesun Park, Ian Davidson, Jackson Mayo, James Bailey, Jimeng Sun, Jingu Kim, Laurent Sorber, Lawrence Carin, Lieven De Lathauwer, Liqing Zhang, Marc Van Barel, Masatoshi Yoshikawa, Matthew Harding, Nguyen Xuan Vinh, Nicholas D. Sidiropoulos, Nico Vervliet, Otto Debals, Papalexakis, Evangelos E., Percy Liang, Petr Tichavsky, Piyush Rai, Prateek Jain, Qibin Zhao, Rafal Zdunekb, Rasmus Bro, Rong Pan, Sammy Hansen, Sewoong Oh, Shun-ichi Amari, Shuo Zhou, Tamara G. Kolda, Tatsuya Yokotaa, Tomoharu Iwata, Volodymyr Kuleshov, Wotao Yin, Xiaomin Fang, Xingyu Wang , Yangyang Xu, Yasuko Matsubara, Yasushi Sakurai, Yu Zhang , Yukihiko Yamashitac, Yunlong He, Yonzhe Jia.

## References

- Claus A Andersson and Rasmus Bro. The n-way toolbox for matlab. *Chemometrics and intelligent laboratory systems*, 52(1):1–4, 2000.
- Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015. URL <http://www.sandia.gov/~tgkolda/TensorToolbox/>.
- J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35(3): 283–319, 1970.
- J Douglas Carroll, Sandra Pruzansky, and Joseph B Kruskal. Candelinc: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*, 45(1):3–24, 1980.
- J Douglas Carroll, Geert De Soete, and Sandra Pruzansky. Fitting of the latent class model via iteratively reweighted least squares candecomp with nonnegativity constraints. In *Multiway data analysis*, pages 463–472. North-Holland Publishing Co., 1989.
- Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

- Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-( $r_1, r_2, \dots, r_n$ ) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000a.
- Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000b.
- Hadi Fanaee-T and João Gama. Eigenevent: An algorithm for event detection from complex data streams in syndromic surveillance. *Intelligent Data Analysis*, 19(3):597–616, 2015a.
- Hadi Fanaee-T and João Gama. Multi-aspect-streaming tensor analysis. *Knowledge-Based Systems*, 89:332–345, 2015b.
- Hadi Fanaee-T and João Gama. Event detection from traffic tensors: A hybrid model. *Neurocomputing*, 203:22–33, 2016a.
- Hadi Fanaee-T and João Gama. Tensor-based anomaly detection: An interdisciplinary survey. *Knowledge-Based Systems*, 98:130–147, 2016b.
- Hadi Fanaee-T, Márcia DB Oliveira, João Gama, Simon Malinowski, and Ricardo Morla. Event and anomaly detection using tucker3 decomposition. *arXiv preprint arXiv:1406.3266*, 2014.
- Changwei Hu, Piyush Rai, Changyou Chen, Matthew Harding, and Lawrence Carin. Scalable bayesian non-negative tensor factorization for massive count data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 53–70. Springer, 2015.
- Yong-Deok Kim and Seungjin Choi. Nonnegative tucker decomposition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.
- Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.
- Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *2008 Eighth IEEE international conference on data mining*, pages 363–372. IEEE, 2008.
- Steven Lemm, Benjamin Blankertz, Thorsten Dickhaus, and Klaus-Robert Müller. Introduction to machine learning for brain imaging. *Neuroimage*, 56(2):387–399, 2011.
- Pauli Miettinen. Boolean tensor factorizations. In *2011 IEEE 11th International Conference on Data Mining*, pages 447–456. IEEE, 2011.
- Morten Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):24–40, 2011.



- Morten Mørup, Lars Kai Hansen, Sidse Marie Arnfred, Lek-Heng Lim, and Kristofer Hougaard Madsen. Shift-invariant multilinear decomposition of neuroimaging data. *NeuroImage*, 42(4):1439–1450, 2008.
- Dimitri Nion and Nicholas D Sidiropoulos. Adaptive algorithms to track the parafac decomposition of a third-order tensor. *IEEE Transactions on Signal Processing*, 57(6):2299–2310, 2009.
- Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):16, 2016.
- Yasushi Sakurai. Trimine. Available online, November 2016. URL <http://www.cs.kumamoto-u.ac.jp/~yasushi/src/trimine.zip>.
- Ofek Shilon. Randorthmat.m. Available online, November 2016. URL <https://www.mathworks.com/matlabcentral/fileexchange/11783-randorthmat>.
- Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S Yu, and Christos Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(3):11, 2008.
- Huachun Tan, Yuankai Wu, Guangdong Feng, Wuhong Wang, and Bin Ran. A new traffic prediction method based on dynamic tensor completion. *Procedia-Social and Behavioral Sciences*, 96:2431–2442, 2013.
- Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.
- Paul Tune and Matthew Roughan. Internet traffic matrices: A primer. *Recent Advances in Networking*, 1, 2013.
- Mathuranathan Viswanathan. How to generate awgn noise in matlab/octave(without using in-built awgn function). Available online, June 2015. URL [http://www.gaussianwaves.com/gaussianwaves/wp-content/uploads/2015/06/How\\_to\\_generate\\_AWG](http://www.gaussianwaves.com/gaussianwaves/wp-content/uploads/2015/06/How_to_generate_AWG)
- Qibin Zhao, Guoxu Zhou, Liqing Zhang, Andrzej Cichocki, and Shun-Ichi Amari. Bayesian robust tensor factorization for incomplete multiway data. *IEEE transactions on neural networks and learning systems*, 27(4):736–748, 2016.
- Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating online cp decompositions for higher order tensors. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, pages 1375–1384, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939763. URL <http://doi.acm.org/10.1145/2939672.2939763>.